REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
09-05-2013	Final Report	8-Sep-2008 - 7-Sep-2012

4. TITLE AND SUBTITLE
Mining Program Source Code for Improving Software Quality

5a. CONTRACT NUMBER
W911NF-08-1-0443

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS
Tao Xie

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES
North Carolina State University
Research Administration
2701 Sullivan Drive, Suite 240
Raleigh, NC 27695 -7514

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
54532-CS.48

12. DISTRIBUTION AVAILIBILITY STATEMENT
Approved for Public Release; Distribution Unlimited

13. SUPPLEMENTARY NOTES
The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT
While the last decade has witnessed great advances in assuring high software quality through static verification against software hazards and unexpected behavior, the field has pretty much focused on statically verifying software applications against common properties to detect common programming problems such as incorrect usage of memory pointers. But most quality issues such as correctness, security, and robustness violations are caused by the incorrect usage of application programming interfaces (APIs). Although API properties or behaviors can be

15. SUBJECT TERMS
Defect detection, mining source code, software testing

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Tao Xie
UU	UU	UU	UU		19b. TELEPHONE NUMBER 919-515-3772

Report Title

Mining Program Source Code for Improving Software Quality

ABSTRACT

While the last decade has witnessed great advances in assuring high software quality through static verification against software hazards and unexpected behavior, the field has pretty much focused on statically verifying software applications against common properties to detect common programming problems such as incorrect usage of memory pointers. But most quality issues such as correctness, security, and robustness violations are caused by the incorrect usage of application programming interfaces (APIs). Although API properties or behaviors can be formally specified and statically verified against software applications by the current state-of-the-art static verification tools, these API properties or behaviors are often not documented by the developers, partly due to two major hindrances in practice: manually specifying a large number of properties or behaviors for static verification is often (1) inaccurate or incomplete, and (2) cumbersome and prohibitively expensive. This project develops new approaches that mine program source code for API properties, which are used to conduct static verification on the software application under analysis to detect defects around APIs.

Enter List of papers submitted or published that acknowledge ARO support from the start of the project to the date of this printing. List the papers, including journal references, in the following categories:

(a) Papers published in peer-reviewed journals (N/A for none)

Received		Paper
05/06/2013	21.00	Tao Xie, Suresh Thummalapenta, David Lo, Chao Liu. Data Mining for Software Engineering, IEEE Computer, (08 2009): 35. doi: 10.1109/MC.2009.256
05/09/2013	35.00	Nuo Li, Tao Xie, Maozhong Jin, Chao Liu. Perturbation-based user-input-validation testing of web applications, Journal of Systems and Software, (11 2010): 0. doi: 10.1016/j.jss.2010.07.007
05/09/2013	36.00	Dan Hao, Tao Xie, Lu Zhang, Xiaoyin Wang, Jiasu Sun, Hong Mei. Test input reduction for result inspection to facilitate fault localization, Automated Software Engineering, (08 2009): 0. doi: 10.1007/s10515-009-0056-x
08/31/2011	2.00	Suresh Thummalapenta, Tao Xie. Alattin: mining alternative patterns for defect detection, Automated Software Engineering, (4 2011): 0. doi: 10.1007/s10515-011-0086-z
08/31/2011	1.00	Hao Zhong, Lu Zhang, Tao Xie, Hong Mei. Inferring specifications for resources from natural language API documentation, Automated Software Engineering, (4 2011): 0. doi: 10.1007/s10515-011-0082-3
TOTAL:	**5**	

Number of Papers published in peer-reviewed journals:

(b) Papers published in non-peer-reviewed journals (N/A for none)

Received Paper

TOTAL:

Number of Papers published in non peer-reviewed journals:

(c) Presentations

Number of Presentations: 0.00

Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received Paper

TOTAL:

Number of Non Peer-Reviewed Conference Proceeding publications (other than abstracts):

Peer-Reviewed Conference Proceeding publications (other than abstracts):

Received	Paper	
05/05/2013	15.00	Xusheng Xiao, Amit Paradkar, Suresh Thummalapenta, Tao Xie. Automated extraction of security policies from natural-language software documents, the ACM SIGSOFT 20th International Symposium the Foundations of Software Engineering (FSE 2012). 2012/11/11 00:00:00, Cary, North Carolina. : ,
05/05/2013	20.00	Donia El Kateb, Tejeddine Mouelhi, Yves Le Traon, JeeHyun Hwang, Tao Xie. Refactoring access control policies for performance improvement, the third joint WOSP/SIPEW International Conference on Performance Engineering (ICPE 2012). 2012/04/22 00:00:00, Boston, Massachusetts, USA. : ,
05/05/2013	19.00	Rahul Pandita, Xusheng Xiao, Hao Zhong, Tao Xie, Stephen Oney, Amit Paradkar. Inferring method specifications from natural language API descriptions, 2012 34th International Conference on Software Engineering (ICSE). 2012/06/02 00:00:00, Zurich, Switzerland. : ,
05/05/2013	17.00	JeeHyun Hwang, Tao Xie, Donia El Kateb, Tejeddine Mouelhi, Yves Le Traon. Selection of regression system tests for security policy evolution, the 27th IEEE/ACM International Conference on Automated Software Engineering (ASE 2012). 2012/09/03 00:00:00, Essen, Germany. : ,
05/05/2013	18.00	Tao Xie, Tao Xie. Cooperative Testing and Analysis: Human-Tool, Tool-Tool and Human-Human Cooperations to Get Work Done, 2012 12th IEEE Working Conference on Source Code Analysis and Manipulation (SCAM). 2012/09/23 00:00:00, Riva del Garda, Italy. : ,
05/05/2013	16.00	Xiaoyin Wang, Lu Zhang, Tao Xie, Yingfei Xiong, Hong Mei. Automating presentation changes in dynamic web applications via collaborative hybrid analysis, the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering (FSE 2012). 2012/11/11 00:00:00, Cary, North Carolina. : ,
05/05/2013	14.00	Qian Wu, Guangtai Liang, Qianxiang Wang, Tao Xie, Hong Mei. Iterative mining of resource-releasing specifications, 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2011/11/06 00:00:00, Lawrence, KS, USA. : ,
05/05/2013	13.00	Wujie Zheng, Hao Ma, Michael R. Lyu, Tao Xie, Irwin King. Mining test oracles of web search engines, 2011 26th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2011/11/06 00:00:00, Lawrence, KS, USA. : ,
05/06/2013	22.00	Suresh Thummalapenta, Tao Xie. Alattin: Mining Alternative Patterns for Detecting Neglected Conditions, 2009 24th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2009/11/16 00:00:00, Auckland, New Zealand. : ,
05/06/2013	25.00	Suresh Thummalapenta, Tao Xie, Nikolai Tillmann, Peli de Halleux, Wolfram Schulte. MSeqGen: object-oriented unit-test generation via mining source code, the 7th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2009). 2009/08/24 00:00:00, . : ,
05/06/2013	24.00	Nuo Li, Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Wolfram Schulte. Reggae: Automated Test Generation for Programs Using Complex Regular Expressions, 2009 24th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2009/11/16 00:00:00, Auckland, New Zealand. : ,

05/06/2013 23.00 Hao Zhong, Lu Zhang, Tao Xie, Hong Mei. Inferring Resource Specifications from Natural Language API Documentation,
2009 24th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2009/11/16 00:00:00, Auckland, New Zealand. : ,

05/09/2013 27.00 Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Wolfram Schulte. Fitness-guided path exploration in dynamic symbolic execution,
The 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009). 2009/06/29 00:00:00, Lisbon, Portugal. : ,

05/09/2013 47.00 Michael Gegick, Pete Rotella, Tao Xie. Identifying security bug reports via text mining: An industrial case study,
2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010). 2010/05/02 00:00:00, Cape Town, South Africa. : ,

05/09/2013 46.00 Hao Zhong, Suresh Thummalapenta, Tao Xie, Lu Zhang, Qing Wang. Mining API mapping for language migration,
the 32nd ACM/IEEE International Conference on Software Engineering (ICSE). 2010/05/01 00:00:00, Cape Town, South Africa. : ,

05/09/2013 45.00 Lingming Zhang, Tao Xie, Lu Zhang, Nikolai Tillmann, Jonathan de Halleux, Hong Mei. Test generation via Dynamic Symbolic Execution for mutation testing,
2010 IEEE 26th International Conference on Software Maintenance (ICSM). 2010/09/12 00:00:00, Timi oara, Romania. : ,

05/09/2013 44.00 Rahul Pandita, Tao Xie, Nikolai Tillmann, Jonathan de Halleux. Guided test generation for coverage criteria,
2010 IEEE 26th International Conference on Software Maintenance (ICSM). 2010/09/12 00:00:00, Timi oara, Romania. : ,

05/09/2013 43.00 Kunal Taneja, Nuo Li, Madhuri R. Marri, Tao Xie, Nikolai Tillmann. MiTV: multiple-implementation testing of user-input validators for web applications,
the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010). 2010/09/20 00:00:00, Antwerp, Belgium. : ,

05/09/2013 42.00 Kunal Taneja, Yi Zhang, Tao Xie. MODA: automated test generation for database applications via mock objects,
the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010). 2010/09/20 00:00:00, Antwerp, Belgium. : ,

05/09/2013 41.00 Guangtai Liang, Ling Wu, Qian Wu, Qianxiang Wang, Tao Xie, Hong Mei. Automatic construction of an effective training set for prioritizing static analysis warnings,
the 25th IEEE/ACM International Conference on Automated Software Engineering (ASE 2010). 2010/09/20 00:00:00, Antwerp, Belgium. : ,

05/09/2013 40.00 Tao Xie, Jonathan de Halleux, Nikolai Tillmann, Wolfram Schulte. Teaching and training developer-testing techniques and tool support,
the 25th Annual ACM Conference on Systems, Programming, Languages, and Applications: Software for Humanity (SPLASH 2010), Educators' and Trainers' Symposium. 2010/10/17 00:00:00, Reno/Tahoe, Nevada, USA. : ,

05/09/2013 39.00 Xiaoyin Wang, Lu Zhang, Tao Xie, Hong Mei, Jiasu Sun. Locating need-to-translate constant strings in web applications,
the 18th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE 2010). 2010/11/07 00:00:00, Santa Fe, New Mexico, USA. : ,

05/09/2013 38.00 Ahmed E. Hassan, Tao Xie. Software intelligence: the future of mining software engineering data,
FSE/SDP Workshop on the Future of Software Engineering Research (FoSER 2010). 2010/11/07 00:00:00, Santa Fe, New Mexico, USA. : ,

05/09/2013 37.00 Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Wolfram Schulte. Future of developer testing: building quality in code,
FSE/SDP Workshop on the Future of Software Engineering Research (FoSER 2010). 2010/11/07 00:00:00, Santa Fe, New Mexico, USA. : ,

05/09/2013 34.00 Suresh Thummalapenta, Tao Xie. SpotWeb: Detecting Framework Hotspots and Coldspots via Mining Open Source Code on the Web,
2008 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE 2008). 2008/09/15 00:00:00, L'Aquila, Italy. : ,

05/09/2013 33.00 Mithun Acharya, Tao Xie. Mining API Error-Handling Specifications from Source Code,
the 12th International Conference on Fundamental Approaches to Software Engineering (FASE 2009). 2009/03/22 00:00:00, . : ,

05/09/2013 32.00 Xiaoyin Wang, Lu Zhang, Tao Xie, Hong Mei, Jiasu Sun. TranStrL: An automatic need-to-translate string locator for software internationalization,
2009 IEEE 31st International Conference on Software Engineering, Formal Demonstration. 2009/05/16 00:00:00, Vancouver, BC, Canada. : ,

05/09/2013 31.00 Kunal Taneja, Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Wolfram Schulte. Guided path exploration for regression test generation,
2009 31st International Conference on Software Engineering - Companion Volume. 2009/05/16 00:00:00, Vancouver, BC, Canada. : ,

05/09/2013 30.00 Wujie Zheng, Michael R. Lyu, Tao Xie. Test selection for result inspection via mining predicate rules,
2009 31st International Conference on Software Engineering - Companion Volume. 2009/05/16 00:00:00, Vancouver, BC, Canada. : ,

05/09/2013 29.00 Xiaoyin Wang, Lu Zhang, Tao Xie, Hong Mei, Jiasu Sun. Locating need-to-translate constant strings for software internationalization,
2009 IEEE 31st International Conference on Software Engineering (ICSE 2009). 2009/05/16 00:00:00, Vancouver, BC, Canada. : ,

05/09/2013 28.00 Suresh Thummalapenta, Tao Xie. Mining exception-handling rules as sequence association rules,
2009 IEEE 31st International Conference on Software Engineering (ICSE 2009). 2009/05/16 00:00:00, Vancouver, BC, Canada. : ,

05/09/2013 26.00 Hao Zhong, Tao Xie, Lu Zhang, Jian Pei, Hong Mei . MAPO: Mining and Recommending API Usage Patterns,
The 23rd European Conference on Object-Oriented Programming (ECOOP 2009). 2009/07/06 00:00:00, . : ,

08/31/2011 3.00 Tao Xie, Nikolai Tillmann, Jonathan de Halleux, Zhendong Su, Suresh Thummalapenta. Synthesizing Method Sequences for High-Coverage Testing,
Proceedings of ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications(OOPSLA 2011). 2011/10/21 12:00:00, . : ,

08/31/2011 4.00 Kunal Taneja, Mark Grechanik, Rayid Ghani, Tao Xie. Testing Software In Age Of Data Privacy: A Balancing Act,
Proceedings of the 8th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE 2011). 2011/09/04 12:00:00, . : ,

08/31/2011 5.00 Kunal Taneja, Tao Xie, Nikolai Tillmann, Jonathan de Halleux. eXpress: Guided Path Exploration for Efficient Regression Test Generation,
the 2011 International Symposium on Software Testing and Analysis. 2011/07/16 12:00:00, Toronto, Ontario, Canada. : ,

08/31/2011 6.00 Yitao Ni, Lu Zhang, Zhongjie Li, Tao Xie, Hong Mei. Detecting Concurrency-Related Problematic Activity Arrangement in WS-BPEL Programs,
the 8th International Conference on Services Computing(SCC 2011). 2011/07/03 12:00:00, . : ,

08/31/2011 7.00 Tao Xie, Xusheng Xiao, Nikolai Tillmann, Jonathan de Halleux. Precise identification of problems for structural test generation,
Proceeding of the 33rd International Conference on Software Engineering (ICSE). 2011/05/20 12:00:00, Waikiki, Honolulu, HI, USA. : ,

08/31/2011 8.00 Suresh Thummalapenta, Madhuri Marri, Tao Xie, Nikolai Tillmann, Jonathan de Halleux. Retrofitting Unit Tests for Parameterized Unit Testing ,
Proceedings of International Conference on Fundamental Approaches to Software Engineering(FASE 2011). 2011/03/25 12:00:00, . : ,

08/31/2011 9.00 Lin Shi, Hao Zhong, Tao Xie, Mingshu Li. An Empirical Study on Evolution of API Documentation,
Proceedings of International Conference on Fundamental Approaches to Software Engineering (FASE 2011). 2011/03/25 12:00:00, . : ,

08/31/2011 10.00 Xusheng Xiao, Tao Xie, Nikolai Tillmann, Jonathan de Halleux. Covana: Precise Identification of Problems in Pex,
Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011), Demonstration. 2011/05/20 12:00:00, Waikiki, Honolulu, HI, USA. : ,

08/31/2011 11.00 Xi Ge, Kunal Taneja, Tao Xie, Nikolai Tillmann. DyTa: Dynamic Symbolic Execution Guided with Static Veri?cation Results,
Proceedings of the 33rd International Conference on Software Engineering (ICSE 2011), Demonstration. 2011/05/20 12:00:00, Waikiki, Honolulu, HI, USA. : ,

TOTAL: **41**

Number of Peer-Reviewed Conference Proceeding publications (other than abstracts):

(d) Manuscripts

Received Paper

TOTAL:

Number of Manuscripts:

Books

Received Paper

TOTAL:

Patents Submitted

Patents Awarded

Awards

2012-2013 ASE 2007 Paper being the Most Cited ASE Papers in the Past Five Years, According to Google Scholar Metrics

2011 Microsoft Research Software Engineering Innovation Foundation Award

2010 IBM Faculty Award

2010 NCSU Sigma Xi Faculty Research Award

2009 ASE 2009 Best Paper Award

2009 ACM SIGSOFT Distinguished Paper Award (for an ASE 2009 paper)

2009 National Science Foundation Faculty Early Career Development (CAREER) Award

2009 Inducted into Sigma Xi, the Scientific Research Society

2009 IBM Faculty Award

Graduate Students

NAME	PERCENT_SUPPORTED	Discipline
Rahul Pandita	0.50	
Kunal Taneja	0.75	
Yoonki Song	0.50	
Xusheng Xiao	0.50	
JeehHyun Hwang	0.25	
Suresh Thummalapenta	0.40	
Mithun Acharya	0.20	
FTE Equivalent:	**3.10**	
Total Number:	**7**	

Names of Post Doctorates

NAME	PERCENT_SUPPORTED
FTE Equivalent:	
Total Number:	

Names of Faculty Supported

NAME	PERCENT_SUPPORTED	National Academy Member
Tao Xie	0.04	
FTE Equivalent:	**0.04**	
Total Number:	**1**	

Names of Under Graduate students supported

NAME	PERCENT SUPPORTED	Discipline
Justin Gorham	0.25	Computer and Computational Sciences
FTE Equivalent:	**0.25**	
Total Number:	**1**	

Student Metrics

This section only applies to graduating undergraduates supported by this agreement in this reporting period

The number of undergraduates funded by this agreement who graduated during this period:	1.00
The number of undergraduates funded by this agreement who graduated during this period with a degree in science, mathematics, engineering, or technology fields:	1.00
The number of undergraduates funded by your agreement who graduated during this period and will continue to pursue a graduate or Ph.D. degree in science, mathematics, engineering, or technology fields:	1.00
Number of graduating undergraduates who achieved a 3.5 GPA to 4.0 (4.0 max scale):	0.00
Number of graduating undergraduates funded by a DoD funded Center of Excellence grant for Education, Research and Engineering:	0.00
The number of undergraduates funded by your agreement who graduated during this period and intend to work for the Department of Defense	0.00
The number of undergraduates funded by your agreement who graduated during this period and will receive scholarships or fellowships for further studies in science, mathematics, engineering or technology fields:	0.00

Names of Personnel receiving masters degrees

NAME
Total Number:

Names of personnel receiving PHDs

NAME
Mithun Acharya
Suresh Thummalapenta
Total Number: 2

Names of other research staff

NAME	PERCENT SUPPORTED
FTE Equivalent:	
Total Number:	

Sub Contractors (DD882)

Inventions (DD882)

Scientific Progress

Because software plays a critical role in businesses, governments, and societies, improving software productivity and quality is an important goal of software engineering. Mining software engineering data has recently emerged as a promising means to meet this goal due to two main trends: the increasing abundance of such data and its demonstrated helpfulness in solving numerous real-world problems. Our research on mining SE data in this year has advanced this area in the following primary dimensions.

(1) To assure high quality and security of software systems, our research has contributed a series of new approaches on mining textual software artifacts along the theme of recovering semantic information out of textual software artifacts. In particular, our research [ICSE 2012] developed natural language processing techniques to mine code contracts out of API documents. Our research [FSE 2012] developed natural language processing techniques to mine access control policies from requirements documents. Our contributed research is the first of such kind in the reserach literature.

(2) To address business requirements and to survive in competing markets, companies or open source organizations often have to release different versions of their projects in different languages. Manually migrating projects from one language to another (such as from Java to C#) is a tedious and error-prone task. To reduce manual effort or human errors, tools can be developed for automatic migration of projects from one language to another. However, these tools require the knowledge of how Application Programming Interfaces (APIs) of one language are mapped to APIs of the other language, referred to as API mapping relations.

Our research [ICSE 10] is the first to mine API mapping relations from one language to another using API client code. MAM accepts a set of projects each with two versions in two languages and mines API mapping relations between those two languages based on how APIs are used by the two versions. These mined API mapping relations assist in migration of projects from one language to another. We implemented a tool and conducted two evaluations to show the effectiveness of MAM. The results show that our tool mines 25,805 unique mapping relations of APIs between Java and C# with more than 80% accuracy. The results also show that mined API mapping relations help reduce 54.4% compilation errors and 43.0% defects during migration of projects with an existing migration tool, called Java2CSharp. The reduction in compilation errors and defects is due to our new mined mapping relations that are not available with the existing migration tool.

(3) In order to improve ineffective warning prioritization of static analysis tools, various approaches have been proposed to compute a ranking score for each warning. In these approaches, an effective training set is vital in exploring which factors impact the ranking score and how. While manual approaches to build a training set can achieve high effectiveness but suffer from low efficiency (i.e., high cost), existing automatic approaches suffer from low effectiveness.

Our research [ASE 10a] proposes an automatic approach for constructing an effective training set. In our approach, we select three categories of impact factors as input attributes of the training set, and propose a new heuristic for identifying actionable warnings to automatically label the training set. Our empirical evaluations show that the precision of the top 22 warnings for Lucene, 20 for ANT, and 6 for Spring can achieve 100% with the help of our constructed training set.

(4) A bug-tracking system such as Bugzilla contains bug reports (BRs) collected from various sources such as development teams, testing teams, and end users. When bug reporters submit bug reports to a bug-tracking system, the bug reporters need to label the bug reports as security bug reports (SBRs) or not, to indicate whether the involved bugs are security problems. These SBRs generally deserve higher priority in bug fixing than not-security bug reports (NSBRs). However, in the bug-reporting process, bug reporters often mislabel SBRs as NSBRs partly due to lack of security domain knowledge. This mislabeling could cause serious damage to software-system stakeholders due to the induced delay of identifying and fixing the involved security bugs.

Our research [MSR 10] proposes a new approach that applies text mining on natural-language descriptions of BRs to train a statistical model on already manually-labeled BRs to identify SBRs that are manually-mislabeled as NSBRs. Security engineers can use the model to automate the classification of BRs from large bug databases to reduce the time that they spend on searching for SBRs. We evaluated the model's predictions on a large Cisco software system with over ten million source lines of code. Among a sample of BRs that Cisco bug reporters manually labeled as NSBRs in bug reporting, our model successfully classified a high percentage (78%) of the SBRs as verified by Cisco security engineers, and predicted their classification as SBRs with a probability of at least 0.98.

(5) To improve software quality, static or dynamic verification tools accept programming rules as input and detect their violations in software as defects. As these programming rules are often not well documented in practice, previous work developed various approaches that mine programming rules as frequent patterns from program source code. Then these approaches use static defect-detection techniques to detect pattern violations in source code under analysis. These existing approaches often produce many false positives due to various factors.

Our research [ASE 09a] proposes a novel approach, called Alattin, to reduce false positives produced by these mining

approaches. Alattin includes a new mining algorithm and a technique for detecting neglected conditions based on our mining algorithm. Our new mining algorithm mines alternative patterns in example form “P1 or P2”, where P1 and P2 are alternative rules such as condition checks on method arguments or return values related to the same API method. We conduct two evaluations to show the effectiveness of our Alattin approach. Our evaluation results show that (1) alternative patterns reach more than 40% of all mined patterns for APIs provided by six open source libraries; (2) the mining of alternative patterns helps reduce nearly 28% of false positives among detected violations.

(6) Typically, software libraries provide API documentation, through which developers can learn how to use libraries correctly. However, developers may still write code inconsistent with API documentation and thus introduce bugs, as existing research shows that many developers are reluctant to carefully read API documentation. To find those bugs, researchers have proposed various detection approaches based on known specifications. To mine specifications, many approaches have been proposed, and most of them rely on existing client code. Consequently, these mining approaches would fail to mine specifications when client code is not available.

Our research [ASE 09b] we propose an approach, called Doc2Spec, that infers resource specifications from API documentation. For our approach, we implemented a tool and conducted an evaluation on Javadocs of five libraries. The results show that our approach infers various specifications with relatively high precisions, recalls, and F-scores. We further evaluated the usefulness of inferred specifications through detecting bugs in open source projects. The results show that specifications inferred by Doc2Spec are useful to detect real bugs in existing projects.

This paper received the ASE 2009 Best Paper Award and ACM SIGSOFT Distinguished Paper Award.

(7) An objective of unit testing is to achieve high structural coverage of the code under test. Achieving high structural coverage of object-oriented code requires desirable method-call sequences that create and mutate objects. These sequences help generate target object states such as argument or receiver object states (in short as target states) of a method under test. Automatic generation of sequences for achieving target states is often challenging due to a large search space of possible sequences. On the other hand, code bases using object types (such as receiver or argument object types) include sequences that can be used to assist automatic test-generation approaches in achieving target states.

Our research [ESEC/FSE 09] proposes a novel approach, called MSeqGen, that mines code bases and extracts sequences related to receiver or argument object types of a method under test. Our approach uses these extracted sequences to enhance two state-of-the-art test-generation approaches: random testing and dynamic symbolic execution. We conduct two evaluations to show the effectiveness of our approach. Using sequences extracted by our approach, we show that a random testing approach achieves 8.7% (with a maximum of 20.0% for one namespace) higher branch coverage and a dynamic-symbolic-execution-based approach achieves 17.4% (with a maximum of 22.5% for one namespace) higher branch coverage than without using our approach. Such an improvement is significant as the branches that are not covered by these state-of-the-art approaches are generally quite difficult to cover.

(8) Our research [ASE 08a, ASE 09a, ICSE 09a] is the first to expand the mining scope from one or a few local project code bases (often not sufficient for mining real API properties) to the Internet-scale open source repositories for API property mining. In particular, our research has exploited code search engines such as Google code search to collect a sufficiently large number of client code examples for a specific API under analysis, and mine API properties from these client code examples.

(9) Our research (described in preliminary work [ESEC/FSE 07] for this project) is the first to exploit and adapt advanced data mining techniques (such as partial order mining) to address unique mining requirements (such as expressing properties for APIs as partial orders), which cannot be satisfied by basic mining techniques commonly used by previous research.

(10) Our research [ICSE 09a, ASE 09a] has investigated complex patterns in common types of API properties and contributed new mining techniques to effectively mine these patterns, without being constrained by available mining techniques from the data mining community. In particular, we have developed novel techniques [ICSE 09a] that mine sequence association rules, a new pattern proposed in our research, for expressing exception-handling properties. We have developed novel techniques [ASE 09a] that mine alternative patterns, a new pattern proposed in our research, for detecting neglected conditions. Our research has detected a significant number of real defects in open source projects with these new mined patterns.

The PI is one of leading researchers in actively promoting this area on mining software engineering data within and even outside the SE community. He has constructed and maintained the first and only comprehensive bibliography on mining SE data. He co-presented tutorials or technical briefings on software
analytics or mining software engineering data at top software engineering venues (7 times at ICSE, 1 time at FSE, and 1 time at ASE) and data mining venues (KDD and ICDM). He will co-organize the 2013 NII Shonan Meeting on Software Analytics: Principles and Practice. He co-organized the 2007 Dagstuhl Seminar on Mining Programs and Processes.

Our research has also improved the Microsoft Research Pex tool for testing of Object-Oriented (OO) Software. Testing OO software (such as those written in C#) is critical because OO languages have been increasingly used in developing modern software systems, and assuring these systems' reliability is very important. In unit testing of OO software, one important and yet challenging problem is to generate desirable method sequences to produce specific receiver or argument object states to find bugs or achieve new code coverage. The search space for such desirable method sequences is huge and there existed no previous techniques to effectively address this problem. We have developed various novel techniques for improving the effectiveness of symbolic execution in method-sequence generation. Our MSeqGen technique [ESEC/FSE 09] is the first to use code mining to gather already used method sequences to guide method-sequence generation. We have also collaborated with Microsoft Research on developing techniques for security testing [ASE 10b], database application testing [ASE 10c], advanced coverage criteria [ICSM 10a], mutation testing [ICSM 10b], string operations [ASE 09c].

The PI is one of leading researchers in actively promoting this area of automated software testing. He had co-presented tutorials on automated software testing at top SE venues (ICSE 2009 and 2010, and OOPSLA 2009). He co-organized 2010 Dagstuhl Seminar on Practical Software Testing: Tool Automation and Human Factors.

Technology Transfer